ELSEVIER

Discrete Applied Mathematics 72 (1997) 193–197

DISCRETE APPLIED MATHEMATICS

Communication

A new version of on-line variable-sized bin packing[1]

Guochuan Zhang

Institut für Mathematik, Technische Universität Graz, Austria

Communicated by Rainer E. Burkard

Received 23 January 1996

Abstract

This paper investigates a new version of the on-line variable-sized bin packing problem. Suppose that bin capacities can vary. Given a list of items, the goal is to pack items in the bins which arrive in an on-line way such that the total size of bins used is minimized. It is shown that next fit and first fit (decreasing) algorithms all have a worst-case performance bound of 2.

AMS classification: 90B35, 90C27

Keywords: Bin packing; On-line algorithm; Worst-case analysis

1. Introduction

In the standard version of variable-sized bin packing, we are given a list $L = (a_1, \ldots, a_n)$ of items, each with item size $s(a_i) \in (0, 1]$, and several different types $B^1, \ldots, B^l$ of bins with sizes $1 = s(B^1) > s(B^2) > \cdots > s(B^l)$. There is an inexhaustible supply of bins of each size. The goal is to pack the given items into the bins so that the sum of the sizes of the bins used is minimum.

In the on-line version of variable-sized bin packing studied in several papers [3–5, 8, 9], we cannot preview and rearrange the items of L before packing, but must instead accept and immediately pack each item as it arrives.

In this note we consider a new on-line version of variable-sized bin packing. We have all information on the items, but we cannot preview the types of the bins before packing. We must decide which items should be packed in the bin as it arrives. A new

[1] This research is supported by the Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung.

PII S0166-218X(96)00018-2

bin will arrive after the current bin is closed. A closed bin cannot be opened again. An essential assumption is that each bin size is not less than the size of the largest item. Observe that for the case that all bins are of size one, this is just the classical one-dimensional bin packing problem.

This on-line restriction arises in some applications. Consider a flexible manufacture system. Here the items represent jobs to be performed and the item sizes correspond to the times required to execute the jobs. In a workshop there is a list L of jobs to be executed on a machine. However, the machine is available for L only in some periods of times (bin sizes) and we do not know any information on the subsequent periods. The goal is to minimize the total available time of the machine for executing the jobs in L.

Clearly, this problem is NP-hard. In the following, we state and analyse some efficient approximation algorithms.

For a list L of items and an approximation algorithm A, let $s(A, L)$ denote the total size of bins used by algorithm A and let $s(\mathrm{OPT}, L)$ denote the total size of bins used in an optimal packing. Then the worst-case performance bound of algorithm A is defined as

$$S_A^\infty = \lim_{k\to\infty} \sup_L \{s(A, L)/s(\mathrm{OPT}, L) \mid s(\mathrm{OPT}, L) \geqslant k\}.$$

Especially, for the classical bin packing problem, $s(A, L)$ and $s(\mathrm{OPT}, L)$ are just the total number of bins used by algorithm A and the total number of bins used in an optimal packing, respectively.

We will adapt some approximation algorithms from the classical one dimensional bin packing problem to the new model and investigate their behavior. A first fit decreasing (FFD) algorithm as well as a next fit decreasing (NFD) algorithm can perform as badly as a next fit (NF) strategy from the viewpoint of worst-case analysis.

2. Classical bin packing

In 1984, Coffman et al. [2] gave an excellent survey paper on bin packing problems. Almost all the algorithms for the classical bin packing can be found in their paper. In the following, four famous algorithms for the classical bin packing problem are listed, which will be adapted to the new problem later.

(i) The *Next Fit* (NF) algorithm [6] always puts an item a_i into the current open bin if it fits. If the open bin has no room for a_i, it is closed (never accepts any item again) and a new bin is started (becoming open) by putting a_i in it. This is clearly a fast algorithm and $S_{\mathrm{NF}}^\infty = 2$.

(ii) The *First Fit* (FF) algorithm is an improvement of NF. It can be explained in two ways: One explanation uses an on-line model. When packing a_i, put it in the lowest indexed bin into which it will fit (starting a new bin only if a_i will not fit into any open bins). It means that every open bin is not closed until the packing is finished. Another explanation uses an off-line model. In this case, the list of items must be given in advance. There is only one open bin at a time, but we check the items left in the list

one by one and put the first item which fits in the open bin and repeat this process. Only if no item fits in the open bin, the bin is closed and a new bin is opened. We will use the second explanation in this paper. Johnson et al. [7] proved that $S_{FF}^{\infty} = \frac{17}{10}$.

(iii) The *Next Fit Decreasing* (NFD) algorithm first orders the items so that $s(a_1) \geqslant s(a_2) \geqslant \cdots \geqslant s(a_n)$, and then applies NF to the reordered list. Baker and Coffman [1] showed that NFD is much better than NF and $S_{NFD}^{\infty} = 1.69103\ldots$ holds.

(iv) The *First Fit Decreasing* (FFD) algorithm also sorts the items of the list into nonincreasing order, and then applies FF (the second explanation) to the reordered list. Johnson et al. [7] gave the tight worst-case performance bound $\frac{11}{9}$ for the FFD algorithm.

3. On the new problem

Now, we analyse analogous versions of those four algorithms when they are applied to the new problem. Let A denote any of those algorithms. Let $B_1, \ldots, B_m$ denote the ordered list of m bins containing L as closed by algorithm A. We also use B_i to denote the sum of the item sizes in B_i without causing any confusion.

Theorem 3.1. $s(A, L) < 2s(\mathrm{OPT}, L) + 1$, *for any list* L.

Proof. For $1 \leqslant i \leqslant m-1$, $B_i + B_{i+1} > s(B_i)$. Therefore, we have

$$s(A, L) = \sum_{i=1}^{m} s(B_i) < 2 \sum_{i=1}^{m} B_i - B_1 - B_m + s(B_m)$$

$$\leqslant 2 \sum_{i=1}^{m} s(a_i) + 1 \leqslant 2s(\mathrm{OPT}, L) + 1. \quad \square$$

Now, we give an instance to prove that the bound 2 is tight for an A packing. Before constructing a list, we define numbers ε_i as follows. Suppose that $\delta > 0$. Then

$$\varepsilon_0 = 0, \qquad \varepsilon_i = \frac{(i+1)!}{(i+1)! + 1}\,\delta, \quad i = 1, \ldots, n. \tag{1}$$

It is easy to see that for $i = 1, \ldots, n-2$

$$\varepsilon_n + \varepsilon_0 < 2\varepsilon_1 \quad \text{and} \quad 2\varepsilon_{i+1} > \tfrac{3}{2}\delta.$$

So, we have

$$2\varepsilon_{i+1} = \frac{2[(i+2)!]}{(i+2)! + 1}\,\delta \geqslant \frac{2[(i+1)!]}{(i+1)! + \frac{1}{3}}\,\delta$$

$$> \frac{2[(i+1)!] + 1}{(i+1)! + 1}\,\delta = \delta + \varepsilon_i$$

$$> \varepsilon_n + \varepsilon_i.$$

This implies that (1) fulfills the following inequalities:

$$0 = \varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_n < \delta, \quad \varepsilon_n + \varepsilon_i < 2\varepsilon_{i+1}, \quad i = 0, \ldots, n-2. \tag{2}$$

Theorem 3.2. $S_A^{\infty} = 2$.

Proof. Let n be a positive integer and $\delta < 2^{-(n+2)}$ be a sufficient small positive number. Suppose that ε_i, $i = 1, \ldots, n$ are defined according to (1). The items of the list L will belong to $n+2$ groups. The first group contains only one item of size $\frac{1}{2} + \delta$. Group 2 has two items of size $\frac{1}{2}$. The ith group, for $i = 3, \ldots, n+2$, contains 2^{i-2} items of size $\frac{1}{2}(1 - \varepsilon_{i-2})$. Obviously, the items have a nonincreasing order. The group i of items is denoted by T_i.

Now turn to the bins. The bins have $n+3$ groups which are denoted by $B_1, \ldots, B_{n+3}$, respectively. Each group contains only one type of bins. B_1 contains one bin of size 1. B_2 has one bin of size $\frac{1}{2} + \delta$. For $3 \leqslant i \leqslant n+2$, B_i contains 2^{i-3} bins of size $1 - \varepsilon_{i-2}$. The last group B_{n+3} has 2^n bins of size $1 - \delta$.

Let us consider the optimal packing first. Clearly, T_1 can be packed in B_2, one item fills one bin. T_2 can be placed into B_1, two items fit one bin. T_i will be put into B_i, $3 \leqslant i \leqslant n+2$, each bin contains two items. Every bin used is full and the $(n+3)$th group of bins is not required (never arrives). The total size of bins used is

$$\begin{aligned} s(\mathrm{OPT}, L) &= 1 + (\tfrac{1}{2} + \delta) + \sum_{i=0}^{n-1} 2^i(1 - \varepsilon_{i+1}) \\ &< 2^n + 1. \end{aligned}$$

However, the $(n+3)$th group of bins is used by any A packing. T_1 will be packed into B_1 each bin contains one item and the remaining space of each bin is almost a half and will be wasted. T_2 will be placed into B_2 and B_3, each bin contains one item too. For $i = 3, \ldots, n+2$, T_i fills up B_{i+1}, one item fits one bin. It is observed that the remaining space of each bin in B_i $(2 < i \leqslant n+2)$ is

$$\frac{1}{2} - \frac{2\varepsilon_{i-2} - \varepsilon_{i-3}}{2}$$

and the smallest item $\frac{1}{2}(1 - \varepsilon_n)$ does not fit in such a space according to the inequalities (2). Hence, the total size of bins used by any A packing is

$$\begin{aligned} s(A, L) &= s(\mathrm{OPT}, L) + 2^n(1 - \delta) \\ &> 2^{n+1} - 1. \end{aligned}$$

Therefore,

$$\begin{aligned} \frac{s(A, L)}{s(\mathrm{OPT}, L)} &> \frac{2^{n+1} - 1}{2^n + 1} \\ &\to 2 \quad \text{as } n \to +\infty. \end{aligned}$$

It means that

$$S_A^{\infty} \geqslant 2.$$

Applying Theorem 3.1 finally completes the proof. □

4. Conclusion and future research

In this paper, a new problem on on-line variable-sized bin packing is investigated. It is easy to modify some algorithms of classical bin packing to the new problem, but unfortunately even the analogue of FFD has a worst-case performance bound of 2! Note that in our problem there is only one open bin at a time. It can be extended to a general case that there are at most $k(k \geqslant 1)$ open bins at a time. However, with the list in Theorem 3.2, it is easy to find that the bound 2 cannot be improved for the FFD algorithm even in such a general case. An obvious open problem is to design some algorithms with a worst-case performance bound less than 2. Giving a lower bound for this kind of problem is another interesting work.

References

[1] B.S. Baker and E.G. Coffman, Jr., A tight asymptotic bound for next-fit-decreasing bin packing, SIAM J. Algebraic Discrete Methods 2 (1981) 147–152.
[2] E.G. Coffman Jr., M.R. Garey and D.S. Johnson, Approximation algorithms for bin packing problems: An updated survey, in: Ausiello and Lucertini, eds., Analysis and Design of Algorithms in Combinatorial Optimization (Springer, New York, 1984) 49–106.
[3] J. Csirik, An on-line algorithm for variable-sized bin packing, Acta Inform. 26 (1989) 697–709.
[4] D.K. Friesen and M.A. Langston, Variable sized bin packing, SIAM J. Comput. 15 (1986) 222–229.
[5] G. Galambos and G.J. Woeginger, On-line bin packing – a restricted survey. Z. Oper. Res. 42 (1995) 25–45.
[6] D.S. Johnson, Near-optimal bin packing algorithms, Tech. Report MAC TR109, Project MAC, Massachusetts Institute of Technology, Cambridge, Mass (1973).
[7] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey and R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3 (1974) 299–325.
[8] N.G. Kinnersley and M.A. Langston, Online variable-sized bin packing, Discrete Appl. Math. 22 (1988/89) 143–148.
[9] G.C. Zhang, Worst-case analysis of the FFH algorithm for on-line variable-sized bin packing, Computing, to appear.